社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－OS－101 (9)
2006／2／17

Linuxにおけるストレージシステムフレームワークの実現

藤 田 智 成†

Linuxt target framework (tgt) は，ストレージターゲットドライバのための新しいフレームワークである．tgt は，ストレージプロトコルに非依存であるため，SCSI，AOE，NBD 等の様々な SAN プロトコルに対応することができる．加えて，ネットワークトランスポートプロトコルにも非依存であり，SCSI プロトコルの場合，FCP，iSCSI，SRP 等の SCSI トランスポートプロトコルに対応することができる．tgt のコア API は，全てのターゲットドライバが必要とする，コマンドキューイング，エラー処理，ターゲットとデバイスの管理，統一されたユーザ空間の制御インターフェイス等の共通機能を提供する．プロトコル処理のようにストレージプロトコルに依存する機能は，各プロトコルライブラリによって提供される．ストレージプロトコルに非依存である点と，モジュール型のデザインの採用により，tgt は，新たなストレージプロトコルに容易に対応することができる．

Storage System Framework in Linux

TOMONORI FUJITA†

Linux target framework (tgt) is a new facility that allows large simplifications in any storage target drivers. Because of storage-protocol independence, tgt can be used for various storage area network (SAN) protocols such as SCSI, AOE, and NBD. Tgt is also independent of network transport protocols, thus in the case of SCSI, it can handle any SCSI transport protocol like FCP, iSCSI, SRP, etc. Tgt has a core set of APIs that provides common features needed by every target driver such as queueing commands, error handling, target and pdevice management, and unified user-space interfaces. Protocol dependant features like protocol processing are provided by tgt protocol helper libraries. The storage-protocol independence and the modular design enable tgt to support a new storage protocol easily.

1. はじめに

現在，多くの企業は，計算機とディスクがシステムバスで直接に接続される従来のストレージアーキテクチャ，Direct Attached Storage（DAS）から，計算機とストレージシステム間を高速なネットワークで接続する，Storage Area Network （SAN）と呼ばれるストレージアーキテクチャへ移行しつつある．

SAN を導入することで，複数の計算機に分散していたディスクを，少数のストレージシステムに集約できるため，ディスクスペースの利用効率が向上し，容量の拡大やバックアップ等の管理作業も容易になる．

SAN を構築するためのストレージ☆技術は複数存在する．

現在，主流の SAN 技術は，Fibre Channel（FC）である．FC 専用の HBA とスイッチを使って，計算機とストレージシステムを接続し，Fibre Channel Protocol（FCP）を使って，SCSI コマンドを転送する．

iSCSI[1] は，Ethernet の NIC とスイッチで構築されたネットワーク上で，計算機とストレージシステムが TCP のコネクションを張り，SCSI コマンドを実行する．

InfiniBand（IB）や RNIC 等，RDMA 機能をサポートするハードウェアを利用したネットワークでは，SCSI RDMA Protocol（SRP）又は，iSCSI Extension to RDMA （iSER）プロトコルを使うことで，計算機とストレージシステムは SCSI コマンドを実行できる．

これまで説明したストレージプロトコルは，いずれも SCSI プロトコルを実行するが，SCSI プロトコルを使わないストレージプロトコルもある．そのようなプロトコルには，ATA プロトコルを Ethernet フレームにのせる ATA over Ethernet（AOE）[2] や，TCP/IP コネクション上で独自プロトコルを利用する Network Block Device （NBD）[3] 等がある．

様々なベンダがストレージシステムを販売しているが，汎用の計算機とオープンソースのオペレーティン

† NTT サイバーソリューション研究所
NTT Cyber Solutions Laboratories

☆ 以降，本稿では特に断らない限り，ストレージとはネットワークストレージを意味する．

グシステムを利用することで，ストレージシステムを実現することもできる．このようなストレージシステムは，専用のハードウェアとオペレーティングシステムを用いる商用ストレージシステムと比較して，性能面では劣るが，機能を自由に追加変更できる，低コストである等の利点を持つ．

Linux では，上記のストレージプロトコルの幾つかに関しては，ストレージシステムを構築するためのソフトウェアが存在する．しかし，各々のソフトウェアがストレージシステムに共通する機能を独自に実装しているため，標準カーネルに取り込んだ場合，コードの重複が発生する，という問題がある．

本稿では，筆者が開発を進めている，ストレージプロトコルの種類に依存しない，Linux を使ったストレージシステム構築用ソフトウェアのフレームワーク，Linux Target Framework（以下，tgt と呼ぶ）について述べる．

本稿の構成は以下の通りである．2 章で設計，3 章で具体的な実装を説明する．4 章で関連研究を述べる．最後に 5 章でまとめる．

2. 設　　計

本稿では，コマンドを発行する計算機をイニシエータ，コマンドに従ってサービスを適用するストレージシステムをターゲットと呼ぶ．

tgt の設計は，以下の要件に基づいている．

複数ストレージプロトコルへの対応　様々なストレージプロトコルとそのハードウェアアダプタに対応可能であること．加えて，同時に複数のストレージプロトコルを動作させることができること．

ディスク仮想化機能の利用　Linux カーネルが提供するディスク仮想化機能を利用し，Logical Volume Manager（LVM）[4] の仮想的ディスクや，ATA，SATA，SCSI，USB 等の様々インターフェイスのディスクをイニシエータに提供可能であること．

統一された管理インターフェイスの提供　ストレージプロトコルに依存しない，統一された管理インターフェイスをユーザ空間に提供すること．

2.1 全体構成

tgt は，ターゲット，イニシエータに加えて，セッション，デバイスという抽象概念を持つ．

tgt は，同時に複数のターゲットを動作させることができ，1 つのターゲットは同時に複数のイニシエータにサービスを提供することができる（ただし，1 つのターゲットがサービスを提供できるイニシエータの数が 1 個に制限されているストレージプロトコルもある）．

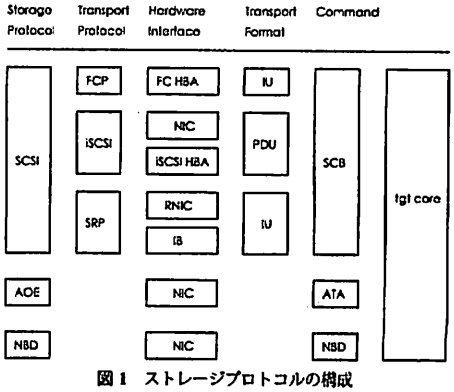

図 1　ストレージプロトコルの構成

セッションは，ターゲットがサービスを提供しているイニシエータを識別するための概念で，1 つのイニシエータに対して 1 つのセッションが存在する．ターゲットが新たなイニシエータにサービスを開始する時に作成される．

デバイスは，ターゲットがイニシエータに提供するサービスである．サービスの種類はプロトコルによって異なるが，SCSI プロトコルの場合，ディスク，テープ，光学ドライブ等の種類がある（本稿では，ディスクを対象とする）．

一つのターゲットは複数のデバイスを持つことができ，一つのターゲットが複数のセッションを持っている場合は，セッション間でそのデバイスは共有される（ただし，1 つのターゲットが 1 つのデバイスしか持つことができないストレージプロトコルもある）．

2.2 複数ストレージプロトコルへの対応

図 1 に，tgt，ストレージプロトコル，ストレージネットワークとの接続口になるハードウェアアダプタの関係を示す．

SCSI プロトコルは階層構造になっており，SCSI コマンドの下位に，トランスポートプロトコルと呼ばれるネットワークのデータ転送を担当するネットワーク層依存部分を扱うプロトコルが位置する．SCSI コマンドを実行するデバイスからは，トランスポートプロトコルに関する情報は隠蔽されている．図には，FCP，iSCSI，SRP の 3 種類のトランスポートプロトコルを示した．

一方，AOE や NBD は，トランスポート層という概念を持っておらず，それぞれ，1 種類のネットワーク層にしか対応していない．

FCP の場合，SCSI コマンドは，IU（Information Unit）と呼ばれるフォーマットでカプセル化される．

ハードウェアアダプタとなる FC HBA が，IU 解析，SCSI コマンドの取出しの大部分を実行する．tgt では，ハードウェアアダプタ依存のソフトウェア部分を，TLLD（target low level driver）と呼ぶ．TLLD は，アダプタと tgt のコア機能をつなぐ役割を持つ．FC HBA の制御方法は，ベンダ毎に異なるため，各ベンダの FC HBA 毎の TLLD が必要となる．

iSCSI プロトコルの場合，ハードウェアアダプタは，一般的な Ethernet 用 NIC か，iSCSI プロトコル専用チップを搭載した NIC（iSCSI HBA と呼ぶ）のどちらかになる．iSCSI プロトコルは，SCSI コマンドを，PDU（Protocol Data Unit）と呼ばれる形式でカプセル化し，それを IP パケットとして送受信する．一般の NIC 用の TLLD は，TCP/IP ソケットから PDU を読み出し，解析し，SCSI コマンドを取り出す．一方，iSCSI HBA は，FC 同様に，ネットワークプロトコル処理，PDU の解析，SCSI コマンドの取出しの大部分を実行する機能を持っている．iSCSI プロトコルでは，一般の NIC 用 TLLD と各ベンダ毎の iSCSI HBA 用 TLLD が必要になる．

SRP では，ハードウェアアダプタとして，RNIC や IB HCA（Host Channel Adapter）とが使われる．FC HBA や iSCSI HBA と同様に，各ベンダのハードウェアアダプタ用の TLLD が必要となる．

AOE は，ハードウェアアダプタとして，一般的な Ethernet 用 NIC を利用し，TLLD は，RAW ソケットからデータを読み出し，ATA コマンドを取り出す．

NBD は，ハードウェアアダプタとして，一般的な Ethernet 用 NIC を利用し，TLLD は，TCP/IP ソケットからデータを読み出し，独自のフォーマットで定義されたコマンドを取り出す．

2.3 ディスク仮想化機能の利用

ターゲットが，イニシエータから受け取ったコマンドを加工せずに，接続されている物理デバイスに転送する設計を選択した場合，ストレージプロトコルに対応したインターフェイスのディスクだけしか利用できない．例えば，iSCSI ターゲットの場合，この設計では，SCSI ディスクのみが利用できる．実際，そのような設計がされたターゲットドライバもある[5]．

tgt は，ストレージプロトコルの種類に関係なく，コマンドを解釈することによって，Linux カーネルのディスク仮想化機能を利用する．この機能によって，商用ストレージシステム同様，柔軟なディスク管理が可能になる．具体的には，tgt は，全てのブロックデバイスを扱うことができるため，Linux カーネルがディスクとして認識する全てのリソースを，イニシエータにサービスすることができる．

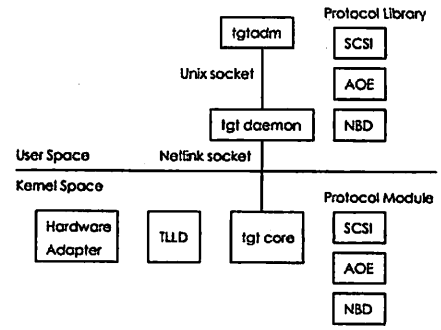

図 2　ソフトウェアコンポーネント構成

各種ストレージプロトコルのコマンドをブロックデバイスにアクセスできるフォーマットに変換する機能はライブラリとして実装されている．TLLD は初期化時にストレージプロトコルを宣言することで，その後の TLLD からのリクエストは，対応するプロトコルのライブラリを使って変換される．プロトコルライブラリは，ディスクに対するリード・ライトコマンドを，ブロックデバイスに対するファイルアクセスに変換する．

2.4 管理インターフェイスの統合

ストレージシステムの管理作業を容易にするために，tgt は，ストレージプロトコルに非依存な管理インターフェイスをユーザに提供する．この機能は，特にマルチストレージプロトコルに対応したストレージシステムで有益である．例えば，1 種類のシェルスクリプトで，ストレージプロトコルに関係なく，全てのターゲットの管理ができる．

前述したように，既存のストレージプロトコルは，ターゲット，イニシエータ，セッション，デバイスという抽象概念でストレージシステムを表わすことができる．従って，その抽象概念に対する操作という形で，ストレージシステム管理も抽象化可能である．例えば，ターゲット，デバイスの追加，削除，オフライン，オンライン操作，セッションの強制削除，また，これらオブジェクトの状態問い合わせ等の操作を tgt は提供する．ストレージプロトコルに依存する管理操作は，プロトコルライブラリによって実現される．

3. 実　　装

3.1 ソフトウェアコンポーネント構成

図 2 と表 1，2 に，tgt のソフトウェアコンポーネント構成をまとめた．

tgt の最も重要な実装方針は，機能の大部分をユーザ空間に実現することである．tgt のカーネルコンポーネ

表 1 カーネル空間ソフトウェアコンポーネント

名称	機能
tgt core	共通機能．ストレージプロトコル非依存，ハードウェアアダプタ非依存機能を提供する．
プロトコルモジュール	ストレージプロトコル依存機能を提供する．ストレージプロトコル毎に用意される．
TTLD	ハードウェアアダプタ依存機能を提供する．

表 2 ユーザ空間ソフトウェアコンポーネント

名称	機能
tgtd	デーモンプログラム．ストレージプロトコル処理，ストレージシステム管理機能を提供する．
tgtadm	統一された管理インタフェースをユーザに提供するユーティリティ．tgtd に接続して，ユーザからの管理要求を伝える．
プロトコルライブラリ	ストレージプロトコル依存機能を提供するダイナミックリンクライブラリ．tgtd がロードする．

ントは，TLLD とユーザ空間を接続し，情報を運ぶ単純なパイプのように働き，主要機能であるイニシエータから届いたコマンドの処理機能はユーザ空間に実装されている．

tgt がこの実装方針に基づいている一番の理由は，Linux カーネルの SCSI サブシステムの責任者が，標準カーネルに採用するための条件[6] の 1 つとしているからである．

カーネル空間で実現されてきたシステムの機能をユーザ空間に実装する手法は，開発作業が容易になる等の利点をもたらすことが知られており，Linux カーネルでも積極的に採り入れられている[7]．

カーネル空間のコンポーネントは，カーネルモジュールとして実現されており，必要なモジュールのみが動的にロードされる．ユーザ空間のコンポーネントは，デーモンプログラムとプロトコルライブラリから構成される．プロトコルライブラリはダイナミックリンクライブラリとして実装されており，必要なライブラリだけがロードされる．

3.2 TLLD

TLLD は，ハードウェアアダプタに直接アクセスする必要があるため（ハードウェア割り込み等），カーネル空間に実装されている．一般的な TLLD の動作は以下の通りである．

(1) 初期化時に，TLLD は，tgt core にストレージプロトコルの種類を宣言する．

(2) 新たにイニシエータにサービスを開始した時には，tgt core にセッションの生成を要求する．

(3) TLLD は，イニシエータからコマンドを受け取り，tgt core に渡す．

(4a) ターゲットがイニシエータにデータを送信するコマンドの場合は，TLLD は，tgt core から，レスポンスが保存されているバッファのアドレスを受け取り，ネットワークに送信するようにハードウェアインターフェイスに命令する．

(4b) イニシエータからターゲットにデータを送信するコマンドの場合は，TLLD は，tgt core から，イニシエータからのデータを保存するバッファのアドレスを受け取り，ハードウェアインターフェイスにバッファの準備ができたことを通知する．ハードウェアインターフェイスはイニシエータにデータを送るように要求し，届いたデータを指定されたバッファに転送する．

(5) ハードウェアインターフェイスがコマンドの処理を終えたら，TLLD は，tgt core にコマンドの完了を通知する．

3.3 tgt core

tgt core は，TLLD からコマンドを受け取り tgtd に転送する，tgtd からレスポンスを受取り TLLD に転送する，という 2 つの役割を持つ．

コマンド受け取り操作では，tgt core は，TLLD から以下の情報を受け取る．

- ターゲットの識別番号
- コマンドが含まれたバッファ（コマンドバッファ）
- コマンドバッファの長さ
- デバイスを特定するためのデータが含まれたバッファ（デバイスバッファ）
- デバイスバッファの長さ
- イニシエータ・ターゲット間でやり取りするデータの長さ
- データの送信方向（イニシエータからターゲット，又は，ターゲットからイニシエータ）

netlink と呼ばれる機能を使って，tgt core と tgtd は接続されている．netlink は，ユーザ空間のプログラムが，ソケット API を使って，カーネル空間とデータをやり取りするための機能である．

tgt core は，プロトコルモジュールの機能を呼び出し，上記の情報をプロトコルに応じた特定のデータフォーマットのパケットにして，tgtd に転送する．

3.4 tgtd

まず最初に，tgtd は，tgt core との間に確立された netlink ソケットから読み出すことで，パケットを 1 つ取り出す．次に，tgtd は，パケットの先頭のターゲットの識別番号から，適切なプロトコルライブラリを判断し，コマンドを処理し，レスポンスを生成する．最後

に，tgtd は，コマンドのレスポンスを，netlink ソケットに書き込むことで，tgt core に伝える．

ユーザ空間とカーネル空間でデータを交換する際の問題の一つは，メモリコピーによる性能低下である．特に，デバイスからの読み書きを実行する I/O コマンドの処理では，ソケットを通じて，大量のデータをコピーすることになる．

tgt では，メモリマップ I/O と呼ばれる手法を使い，メモリコピーを回避している．tgtd は，tgt core とバッファの内容をソケットを通じて送受信するのではなく，バッファをユーザ空間にマップし，tgt core とそのアドレスを送受信する．

I/O コマンドに関しては，tgtd が mmap システムコールを使い，ブロックデバイスファイルのページキャッシュの一部分をユーザ空間に張り付け，mmap システムコールの返り値であるアドレスと張り付けた長さを tgt core に渡す．tgtd は，tgt core からコマンドの終了通知を受け取ると，munmap システムコールを使って，ページキャッシュをアンマップする．

I/O コマンド以外のデバイスからの読み書きが発生しないコマンドの場合，tgtd は，malloc で確保したユーザ空間のバッファにレスポンスを作成し，そのアドレスと長さを tgt core に渡す．コマンドの終了通知を受け取ると，tgtd は確保したメモリを解放する．

メモリマップ I/O は，SCSI generic driver[8] や Direct I/O で使われている手法である．tgt は，カーネル内に既に存在するメモリマップ I/O 用の関数を使い，実装を簡素化することができる．

3.5 プロトコルライブラリ

プロトコルライブラリの最も重要な機能は，イニシエータから届いたコマンドの実行とレスポンスの生成である．

SCSI プロトコルの場合，コマンドの解析は，コマンドが含まれたバッファ（SCSI Control Block）からのコマンド種類と内容の判断，デバイス情報が含まれたバッファからのデバイス番号（Logical Unit Number）の判断，から成り立っている．

プロトコルライブラリが提供する他の機能としては，SCSI プロトコルの Task management functions のような，ターゲット制御機能があげられる．

3.6 tgtadm

tgtadm はストレージシステムの管理にために使われるユーティリティプログラムであり，ストレージプロトコルに依存せずに，統一されたインターフェイスを提供する．

tgtadm は，Unix ドメインソケットを使って tgtd に接続し，管理用の命令を伝え，結果を待ち，ブロックする．tgtd は，対応するプロトコルライブラリを使い，プロトコルに応じた特定のデータフォーマットのパケットにして，netlink ソケット経由で，tgt core に送信する．その後，tgtd は，tgt core から命令に対する結果を受け取り，tgtadm に転送する．

4. 関連研究

SCST（Generic SCSI Target Middle Level）[9] は，Linux を使った SCSI ストレージターゲットドライバのためのフレームワークである．SCSI プロトコルだけを想定している点，統一された管理用インターフェイスを提供しない点が，tgt と異なる．tgt の Qlogic 社 FC HBA 用 TLLD（qla2xxx）は，SCST の実装をベースとしている．

iSCSI Enterprise Target（IET）[10] は，一般的な NIC と Linux を使って，iSCSI ストレージシステムを実現するソフトウェアである．tgt の iSCSI 用 TLLD（istgt）は，IET のコードの一部を利用している．

ibmvscsis[11] は，IBM 社 iSeries の仮想化環境での SRP ストレージシステムを実現するソフトウェアである（RDMA をサポートするハードウェアアダプタを使う SRP ストレージシステムではない）．iSeries の仮想化環境では，VIO サーバと呼ばれる特別な仮想マシンがストレージシステムとなり，残りの仮想マシンがイニシエータとなることで，ストレージ仮想化を実現する．仮想化環境では，RDMA 同様，他の仮想マシンのメモリアドレスを使ったデータ転送が可能である．tgt は，ibmvscsis をベースにした TLLD（ibmvstgt）をサポートする．

SCST，IET，ibmvscsis は，いずれのソフトウェアも，全ての機能がカーネル空間で実現されている点が設計上の誤りとみなされており，これまでのところ，Linux カーネルの標準機能として取り込まれていない．

5. まとめ

本稿では，Linux カーネルを使って，ストレージシステムを実現するためのフレームワーク，tgt について説明した．tgt は，様々なネットワークストレージプロトコルに対応しすることができ，統一された管理用インターフェイスを提供する，という特徴を持つ．可能な限り多くの機能をユーザ空間で実現するという方針に基づいて実装されている．

tgt は性能に大きく影響するストレージプロトコル処理部分をユーザ空間に実装しているため，ユーザ・カーネル空間のメモリコピー等による性能低下が課題

となるが，メモリマップI/Oを用いることで，この問題を解決している．

現在のtgtの性能は，単一のストレージプロトコルのターゲット機能をカーネル内部で実現した，IETの80〜90%となっており，ユーザ空間を使った実装で得られる利点（豊富なライブラリ関数が利用できる，デバッグが容易になる等）を考えると，許容できる性能だと考えている．

tgtは，標準カーネルへの取り込みが予定されており，活発に開発されている状態である．今後は，実装を進める一方，詳細な性能評価を行い，ユーザ空間でのストレージプロトコル処理技法について検討を続けていく予定である．

tgtのソースコードは，`http://developer.berlios.de/projects/stgt/` で入手可能である．

謝辞 共同開発者であるMike Christie，設計に関する有益な助言を下さったChristoph Hellwig，James Bottomleyに感謝の意を表する．

参考文献

1) Satran, J., Meth, K., Sapuntzakis, C., Chadalapaka, M. and Zeidner, E.: Internet Small Computer Systems Interface (iSCSI), RFC 3720 (2004).
2) Coile, B. and Hopkins, S.: The ATA over Ethernet Protocol (2004). `http://www.coraid.com/documents/AoEr8.txt`.
3) Machek, P.: *Network Block Device*. `http://nbd.sourceforge.net/`.
4) Teigland, D. and Mauelshagen, H.: Volume Managers in Linux, *the USENIX Annual Technical Conference*, Boston, MA, pp.185–198 (2001).
5) Chelsio communications: Chelsio T210/T204/T110 Linux Driver. `https://service.chelsio.com/drivers/`.
6) James Bottomley: *stgt a new version of iscsi target?* (2005). `http://www.spinics.net/lists/linux-scsi/msg06101.html`.
7) Goggin, E., Kergon, A., Varoqui, C. and Olien, D.: Linux Multipathing, *Ottawa Linux Symposium*, pp. 147–167 (2005).
8) Douglas Gilbert: *The Linux SCSI Generic (sg) Driver*. `http://sg.torque.net/sg/`.
9) Vladislav Bolkhovitin: Generic SCSI Target Middle Level for Linux (2003). `http://scst.sourceforge.net/`.
10) 藤田智成，小河原成哲：iSCSIターゲットソフトウェアの解析，情報処理学会論文誌：コンピューティングシステム，Vol.46, No.SIG 3 (ACS 8), pp. 38–50 (2005).
11) Boutcher, D. and Engebretsen, D.: Linux Virtualization on IBM POWER5 Systems, *Ottawa Linux Symposium*, pp.113–120 (2004).